Satu Kuosmanen

KORTTILAJITTELUSOVELLUS HTML5-TEKNIIKOIN

Opinnäytetyö
Tietojenkäsittely

Maaliskuu 2011

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

Opinnäytetyön päivämäärä
11.03.2011

Tekijä(t)
Satu Kuosmanen

Koulutusohjelma ja suuntautuminen
Tietojenkäsittely

Nimeke
Korttilajittelusovellus HTML5-tekniikoin

Tiivistelmä

Opinnäytetyöni tarkoituksena on tutustuttaa lukijoille uusi HTML5 ja esitellä muita sitä tukevia tekniikoita, joilla saadaan aikaan dynaamisia verkkosovelluksia. HTML5:den standardi ei ole vielä kokonaan valmis ja selaintuet heikot, mutta onneksi osa selaimia tukee sitä jo. Opinnäytetyössäni haluan myös esitellä muitakin ohjelmointikieliä ja tekniikoita luoda dynaamisia verkkosovelluksia.

Käyttöliittymäsuunnittelu on tärkeä osa verkkosivujen ja sovelluksien tekemisessä. Korttienlajittelu on yksi tapa ratkaista informaatioarkkitehtuuri. Opinnäytetyössäni haluankin tutustuttaa lukijalle, mitä korttilajittelu on.

Tutkimusongelmana on tehdä sovellus, joka mahdollistaa korttilajittelun sekä saada siitä tuleva tieto talteen. Tavoitteena opinnäytetyössä on tehdä sovellus, joka helpottaa sovelluskehittelijöitä tekemään informaatioarkkitehtuuria. Korttienlajittelu tulokset voidaan hakea sovelluksesta Excel-taulukkoon, josta tietoa voidaan analysoida.

Sovellus on toteutettu HTML5-tekniikoita käyttäen. Sovellus on selaimella toimiva ja se helpottaa sekä säästää aikaa ja rahaa käyttöliittymäsuunnitellussa. Saavutin opinnäytetyössäni sille asetetut tutkimusongelmat ja tavoitteet.

Asiasanat (avainsanat)
HTML5, JavaScript, korttilajittelu, RIA,

Sivumäärä	**Kieli**	**URN**
31 s.	Suomi	**http://www.urn.fi/URN:NBN:fi:amk-201103163206**

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi
Janne Turunen

Opinnäytetyön toimeksiantaja
Mikkelin ammattikorkeakoulu / CampusIT

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

Date of the bachelor's thesis
11 March 2011

Author(s)
Satu Kuosmanen

Degree programme and option
Business Information Technology

Name of the bachelor's thesis
Card sorting application with HTML5 methods

Abstract

The meaning of my bachelor's thesis was to make the new HTML5 better known and to introduce other techniques that support it and enable dynamic network applications. The standard of HTML5 was not fully ready yet and the support for web browsers has been weak, but luckily some web browsers supported it already. My thesis also introduced some other programming languages and techniques for making dynamic network applications. User interface design is an important part of making websites and network applications. Card sorting is one way to accomplish information architecture. My thesis explained what card sorting is.

The research problem was to make an application which would make card sorting possible and also save the resulting information. The goal was to make an application that would help application developers in making information architectures. Card sorting results can be moved from an application to Excel worksheet where the information can be analyzed. The application was executed with HTML5 techniques. The application works on a web browser and it helps in user interface design and also saves time and money. The study reached its goals.

Subject headings, (keywords)
HTML5, JavaScript, Card sorting, RIA

Pages	Language	URN
31 p.	Finnish	**http://www.urn.fi/URN:NBN:fi:amk-201103163206**

Remarks, notes on appendices

Tutor	Bachelor's thesis assigned by
Janne Turunen	Mikkeli University of Applied Sciences / CampusIT

SISÄLTÖ

1 JOHDANTO

HTML5:den standardi on vielä työn alla ja ei ole vielä kokonaan valmis, silti osa sen toiminnallisuuksia ja ominaisuuksia toimii joissakin suuremmissa selainten valmistajien selaimissa. Opinnäytetyön tarkoituksena on tutustuttaa lukijaa hieman ja esitellä, mitenkä uudella HTML5:llä voidaan tehdä työpöytä sovelluksia muistuttavia ratkaisuja. Lisäksi opinnäytetyössäni on esiteltynä muita ohjelmointikieliä ja tekniikoita, joilla voidaan saada aikaan samankaltaisia sovelluksia.

Verkkosovellusten kehittäjillä ja tekijöillä on monesti ongelmia saada aikaan käyttäjiä parhaiten palveleva navigaatiorakenne. Onneksi apua saadaan erilaisista käyttöliittymäsuunnittelu tavoista. Korttilajittelu on yksi oivallinen tapa saada selville, millainen olisi käyttäjien mielestä paras mahdollinen navigaatiorakenne.

Opinnäytetyön tutkimusongelmana on toteuttaa sovellus, joka mahdollistaa korttilajittelun HTML5:den avulla, jolla saadaan myös tieto talteen. Sovelluksen tavoitteena on helpottaa sovelluskehittäjien informaatioarkkitehtuurin tekemistä. Eli tutkimusongelmana ja tavoitteena opinnäytetyössäni on tarjota ratkaisu navigaatiorakenteen tekemiseen.

Opinnäytetyön toisessa luvussa on käsiteltynä monia eri tekniikoita toteuttaa rikkaita Internet-sovelluksia. Kolmannessa luvussa olen kertonut, mitä korttilajittelu on sekä miksi ja mihin sitä käytetään. Itse sovelluksen toteutus on käsiteltynä neljännessä luvussa. Luvussa neljä esitellään tekniikka, jolla korttilajittelu on mahdollista ja tapa millä lailla tieto saadaan talteen tietokantaan.

2 RIKKAAT INTERNET-SOVELLUKSET

Käyttäjälle halutaan tarjota parasta mahdollista kokemusta verkkosivuilla ja antaa mahdollisuus samankaltaisiin toimintoihin kuin työpöydällä. RIA on termi, joka tulee sanoista Rich Internet Applications (rikkaat internet-sovellukset) ja sen tarkoitus on tarjota rikasta ja kiinnostavaa kokemusta, joka saa käyttäjän tyytyväiseksi. (Adobe 2011.)

RIA-tekniikoilla tehtyjä sovelluksia käytetään selaimella ja niiden tarkoitus on tarjota käyttäjälle mahdollisuus vuorovaikutukseen, eli dynaamisuuteen. RIA-tekniikalla tehdyt sovellukset ovat nopeita, koska sivua ei tarvitse päivittää, että jokin tapahtuma näkyisi käyttäjälle. Vedä ja pudota -tekniikka, liukusäätimet, animaatiot, sisällön liikuttaminen tai venyttäminen sekä reaaliaikaisesti päivittyvät kaaviot ovat kaikki mahdollisia RIA-tekniikalla tehtynä. (Uoma Oy 2011.)

RIA-tekniikoita on useita ja näitä ovat esimerkiksi Flash, JavaFX ja Silverlight. Kolmella edellä mainitulla on yhteistä se, että kaikki tarvitsevat liitännäisohjelman toimiakseen käyttäjän selaimessa. Dynaamisia verkkosivuja voidaan saada aikaan myös muillakin toteutus tekniikoilla kuten Ajaxilla, DHTML:llä ja HTML5:llä. (Adobe 2011; Microsoft Corporation 2011; Oracle Corporation 2010; W3schools 2010b; W3schools 2011b; W3schools 2011e.)

2.1 HTML5

Termi HTML koostuu sanoista Hyper Text Markup Language, ja nimensä mukaisesti se on merkkauskieli eikä ohjelmointikieli. HTML käyttää merkkaustageja luodakseen verkkosivuja. HTML-tagit ovat niin sanottuja avainsanoja, jotka on ympyröity kulmasulkeilla ja normaalisti HTML-tagit tulevat pareittain. Ensimmäinen tagipari aloittaa ja viimeinen tagipari lopettaa, esimerkiksi <html> ja </html>. Lopetustagin yhteydessä on hyvä muistaa kenoviiva ennen avainsanaa. (W3schools 2011a.)

Korpela ja Linjama (2005, 431) kertovat, että HTML:ää alettiin kehittämään, sekä käyttämään 1990-luvun alussa. HTML:n toinen versio oli ensimmäinen virallinen hyväksytty määrittely, tämä tapahtui vuonna 1995. Versioiden kehitys jatkui ja HTML:n neljäs versio hyväksyttiin vuoden 1997 lopussa. HTML:n versio neljä on ollut pitkään käytössä ja sen muokattu versio 4.01 julkaistiin vuonna 1999. Vasta vuonna 2006 World Wide Web Consortium (W3C) ja Web Hypertext Application Technology Working Group (WHATWG) alkoivat tehdä yhteistyötä luodakseen HTML:stä uuden, viidennen version (W3schools 2011b). Järvinen (2010, 48) kirjoittaa artikkelissaan, että W3C odottaa vuoteen 2022 kaiken olevan vasta valmista, koska uusia tekniikoita ja mahdollisuuksia on niin paljon. Järvinen jatkaa vielä

kertomalla, että monet suurimmat selain versiot tukevat jo moniakin HTML5-tekniikoita.

HTML:ssä käytetään elementtejä, joita voidaan myös kutsua tageiksi, joiden avulla verkkosivulle kerrotaan miten sivun tulee rakentua. Elementtejä voidaan kirjoittaa sisäkkäin ja tämä on olennainen osa HTML:ää. Lohko- ja sisäelementtejä käytetään varsinaisessa rungossa ja ne ovat kaksi niin sanottua pääryhmää. Lohkoelementit pitävät sisällään kappaleita esimerkiksi <p>-tagilla tai -tagilla merkittynä tai taulukoita ja listoja sekä otsikoita. Otsikoilla, kappaleilla, listoilla ja taulukoilla rakennetaan yleensä sivun runko ja muut lohkoelementit täydentävät rakennetta. Sisäelementit taas pitävät sisällään muun muassa kuva elementin ja lomakkeiden kentissä tunnetut input -elementit. (Korpela & Linjama 2005, 72 – 74.)

Uusia elementtejä parempaan, helpompaan ja tarkempaan rakenteen tekemiseen tarjotaankin HTML5:ssä. Esimerkiksi <article>-tagi tarjoaa mahdollisuuden kirjoittaa sisäänsä ulkoista tekstiä blogeista, foorumeilta tai vaikka uutisartikkeleista. Uudella <header>-tagilla voidaan esimerkiksi esitellä sivu tai kappaleosio ja <footer>-tagin sisään sivun lopussa voidaan mainita tekijänoikeus tai tekijä tiedot. (W3schools 2011b.)

HTML5 tarjoaa myös uusia media elementtejä, joilla saadaan esimerkiksi ääniä ja videoita helposti osaksi verkkosivua. Media elementit syrjäyttävät esimerkiksi Flashin kaltaiset liitännäisohjelmat. Tällä hetkellä video elementille on olemassa kaksi eri videoformaattia ja äänielementille kolme eri ääniformaattia, joita ne tukevat. (W3schools 2011b.) Järvinen (2010, 48) kertoo artikkelissaan, että videon halutaan toimivan yhtä varmasti kuin, nykyään jpeg- ja png-kuvat toimivat. Myöskin lomakkeille on tullut uusia elementtejä, joilla voidaan helpommin kontrolloida lähtevää tietoa serverille (W3schools 2011b).

Niin sanottuja tyhjiäkin elementtejä on olemassa, Korpela ja Linjama (2005, 72) mainitsevat muutaman tutun näistä. Rivin vaihtoa varten kirjoitetaan
-tagi, jossa ei ole lopetustagi paria ollenkaan. Nimitys johtuu siitä, ettei tagilla ole sen ihmeempää sisältöä. Toinen tuttu tällainen on <img>-tagi, jolla ei myöskään ole lopetus paria, tällä tagilla saadaan kuvia osaksi verkkosivua. <img>-tagi eroaa kuitenkin
-tagista sen

verran, että <img>-tagin sisälle täytyy kirjoittaa attribuutteja, jotka kertovat mistä kuva löytyy eli osoite sekä vaihtoehtoteksti, jos kuva ei jostain syystä näy verkkosivuilla. Teague (2004, 21) muistuttaa, että XHTML:ssä tyhjät elementit tarvitsevat kenoviivan merkiksi siitä, että se sulkee itse itsensä, esimerkiksi <br />.

Ensimmäinen asia joka HTML5 dokumentissa tulee olla on doctype-määrittely, tällä kerrotaan selaimelle, mikä HTML määrittely on kyseessä, muutoin selain ei ymmärrä millainen dokumentti on kyseessä. HTML4:ssä doctype-määrittelyn yhteydessä vaaditaan viittaus DTD:hen (Document Type Definition), koska HTML4 pohjautuu SGML:ään. HTML5 ei pohjaudu SGML:ään, joten viittausta ei tarvitse lisätä doctype-määrittelyn yhteyteen. Dokumenttityypin määrittely HTML4:ssä ja HTML5:ssä eroaa myös siinä, että HTML4:ssä mahdollisuuksia oli kolme erilaista ja HTML5:ssä vain yksi: <!DOCTYPE HTML>. <!DOCTYPE>-tagilla ei ole lopetustagia eikä se ole merkkikokoriippuvainen. (W3schools 2011b.)

Järvinen artikkelissaan (2010, 50) sanoo, että HTML5 tarjoaa sellaisia ominaisuuksia, joita on ollut vain käyttöjärjestelmissä. Tällaisia ominaisuuksia ovat esimerkiksi vedä ja pudota -toiminto, joka mahdollistaa sen, että dataa voidaan vetää selaimesta tai sovelluksesta toiseen. Näin HTML5 tuo selaimen askeleen lähemmäs käyttöjärjestelmien työpöytämallia.

HTML5 tarjoaa uusia globaaleja attribuutteja, kuten draggable-attribuutin. Draggable-attribuuttia voidaan käyttää yhdessä uuden drag & drop (vedä ja pudota) API:n kanssa. (Kesteren 2010b.) Elementti jolle on annettu draggable-attribuutti, voidaan ohjata tapahtuma attribuuteilla. HTML5 tarjoaakin monia uusia hiirellä tehtäviä tapahtuma attribuutteja, joilla saadaan aikaan mahdollinen elementin vetäminen ja pudottaminen. Videoita, kuvia ja ääniä pystytään ohjaamaan media tapahtuman käsittelijöillä. (W3schools 2011b.) Järvinen (2010, 50) huomauttaa, että JavaScriptillä ohjataan kaikkia HTML5 ominaisuuksia.

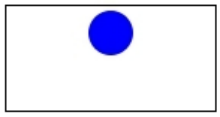
Videoelementin saa osaksi verkkosivua esimerkiksi näin: <video src="movie.ogg" width="320" height="240" controls="controls"></video>. Ensiksi määritellään videoelementti, jollekka on kerrottu mistä videon löytää eli osoite sekä annettu koko

ja annettu vielä controls-attribuutti. Controls-attribuutilla saadaan, esimerkiksi sellaiset nappulat kuten Play(toista) osaksi videota. (W3schools 2011b.)

Verkkosivuilla esiintyvä grafiikka on yleensä bittikarttaa, mutta vuonna 2001 W3C julkaisi standardin vektorigrafiikalle. Scalable vector graphics eli SVG on mahdollistanut kuvien portaallisen suurentamisen, toisin kuin bittikarttakuvat. HTML5:den tarjoama canvas-elementti mahdollistaa grafiikan piirtämisen verkkosivuille uudella <canvas>-tagilla. (Järvinen 2010, 48-50.) Canvas on suorakulmaisen muotoinen alue, jossa pystytään kontrolloimaan jokaista pikseliä. Canvas-elementtillä ei itsessään ole mitään piirto ominaisuuksia, vaan määritelmät piirroksesta on tehtävä JavaScriptillä. Canvas-elementillä voidaan piirtää esimerkiksi viivoja, ympyröitä, laatikoita tai sen sisälle voidaan laittaa kuva. (W3schools 2011b.)

Kuvassa 1 on esitelty miten canvas-elementtiä käytetään HTML5:ssä. Elementille tulee antaa id, jotta JavaScript osaa käsitellä oikeaa elementtiä. Canvas-elementille on annettu koko ja sen ääriviivat on mustalla yhden pikselin paksuisesti piirretty. JavaScriptillä kerrotaan, että sisälle piirretään ympyrä, jonka täyttövärinä on sininen. Ympyrälle määritetään koordinaatit sekä koko. Kuvassa 1 koodi esimerkin alla on näkyvissä, mitä sillä saadaan aikaan.

```
<!DOCTYPE HTML>
<html>
<body>

<canvas id="esimCanvas" width="140" height="70" style="border:1px solid #000000;"> </canvas>

<script type="text/javascript">

var c=document.getElementById("esimCanvas");
var cxt=c.getContext("2d");
cxt.fillStyle="#0000FF";
cxt.beginPath();
cxt.arc(70,18,15,0,Math.PI*2,true);
cxt.closePath();
cxt.fill();

</script>

</body>
</html>
```



KUVA 1. Canvas-elementti esimerkki

Web storage on tapa HTML5:ssä, jolla pystytään tallentamaan tietoa paikallisesti. Tapoja tallentaa tietoa on HTML5:ssä kaksi, localStorage ja sessionStorage. Aiemmin tiedon tallennus tapahtui Cookieilla eli kekseillä, mutta se ei sovellu hyvin tallentamaan isoja määriä tietoja. Tiedon tallentamiseen ja käsiksi pääsemiseen HTML5:ssä käytetään JavaScriptiä. Ensimmäisellä metodilla localStoragella pystytään tallentamaan tietoa, joka on saatavilla seuraavana päivänä tai vaikka vuodenkin päästä. Toinen metodi sessionStorage taas tuhoaa tiedon, kun käyttäjä sulkee selaimen. (W3schools 2011b.)

2.2 DHTML

DHTML termi muodostuu sanoista Dynamic HTML, eli dynaaminen HTML. Tämä tekniikka tuli verkkosovellusten kehittäjille mahdolliseksi käyttää, kun Microsoft ja Netscape julkaisivat selaimistaan neljännen versionsa. DHTML tekniikka koostuu yhdistelmänä HTML:stä, CSS-tyylisivuista, JavaScriptistä sekä DOM-mallista. Näiden neljän tekniikan yhdistäminen antaa kehittäjille sen mahdollisuuden, että he pystyvät sivuston suorituksen aikana muokkaamaan verkkosivun sisältöä ja sen rakennetta. (Asleson & Schutta 2007, 11.)

Teague (2004, 194) kertoo, ettei miksikään koodauskieleksi DHTML:ää voida kutsua, koska se perustuu niin monesta eri asiasta, kyseessä on siis pelkkä termi. Tegue myös jatkaa (2004, 197 – 198) DHTML:n historiasta sen, että kahdella yrityksellä oli omia ideoita siitä millaisia tekniikoita tulisi käyttää, jotta DHTML:stä tulisi entistäkin dynaaminen. Netscapella oli paljon uudenlaisia tekniikoita tarjottavana DHTML:ää varten, mutta valitettavasti CSS pystyi samoihin asioihin, joten niistä ei koskaan rakentunut standardia. Netscape esitteli vaihtoehdon CSS:lle, nimeltään JavaScript style sheet (JSS), joka käyttää JavaScriptin mukaista syntaksia. Netscapen selain versio neljä on ainoa joka tukee JSS:ää, joten tätä tekniikkaa ei suositella käyttämään. Toinen yritys oli Microsoft joka halusi rakentaa DHTML:n pohjautumaan ActiveX teknologian kaltaiseksi. Kuitenkaan sekään ei ole hyvä vaihtoehto, koska se ei toimisi missään muissa, kuin Internet Explorer-selaimissa. Toisena lisäyksenä Microsoft toi DHTML:ään visual filtersit joilla voi luoda visuaalisia efektejä esimerkiksi tekstiin. Ongelmana tässä tuli se, ettei se ollut standardi useimmissa selaimissa, ei edes Microsoftin omissa.

Miksi käyttää DHTML:ää kun on muitakin tekniikoita, joilla ajaa samanlaisia asioita? Tietenkin DHTML omaa joitakin sellaisia etuuksia, jotka puoltavat sen käyttöä. DHTML on ollut joko kokonaan tai osittain tuettu suurimpien selaimien kuten Netscapen ja Internet Explorerin neljänsistä versioista lähtien. Myöskin DHTML on standardoitu tekniikkaa, joita selain valmistajat pystyvät käyttämään, mahdollistaa se sen, että verkkosivu näyttää samalta selaimesta riippumatta. Selkein etu on tietenkin se, että verkkosivua pystytään muokkaamaan lennosta, ilman että sivua pitää päivittää. Tämä tekniikka on nopea ja helppo oppia sekä toteuttaa, eikä se vie paljon tilaa eikä se tarvitse toimiakseen selaimissa minkäänlaista ladattavaa lisäohjelmaa. (Teague 2004, 199 – 200.)

Kuitenkaan DHTML ei ole aina yhtä ruusuilla tanssimista. Teague (2004, 201) antaa näytteitä siitä, missä kohtaa DHTML:llä on heikkouksia. Yhteensopivuus on ongelmallista erilaisissa selaimissa, joskus jopa ongelmia voi aiheuttaa käyttöjärjestelmän yhteensopivuus. CSS ja JavaScript on hyvin tarkkoja siitä onko kaikki oikein, esimerkiksi kaikki tagit tulee olla suljettuna oikein, muuten ne aiheuttavat ongelma tilanteita verkkosivuilla. Sekä ongelmana DHTML:n toimivuudelle voi olla selainten omat viat tai puutteet, jotka voi estää DHTML:ää toimimasta.

2.3 CSS

CSS lyhenne muodostuu sanoista Cascading Style Sheets. Ennen kuin CSS määrityksiä oli edes luotu, oli tarkka elementtien sijoittelu mahdotonta, sekä ulkoasua ei pystytty merkittävästi muokkaamaan. World Wide Web -järjestö, eli W3C, loi lopulta CSS-määritykset. Näiden määrittelyiden avulla suunnittelijat pystyvät luomaan sekä määrittelemään verkkosivujen elementtien ulkoasun, tyypin sekä sijoittamaan elementin halutulle paikalle. CSS mahdollistaa suunnittelijalle helpon ja nopean tavan muokata verkkosivustoa. (Negrino & Smith 2007, 263.)

Versioita CSS:stä on olemassa kolme, joista ensimmäinen luotiin vuonna 1996. Kuitenkin verkkosivujen tekijät huomasivat puutteita CSS:n ensimmäisessä versiossa, niinpä ennen kuin CSS:n toinen versio oli valmis, julkaistiin CSS-P jolla

mahdollistettiin verkkosivuille elementtien tarkka sijoittelu. Standardiksi tuli vuonna 1998 CSS:n toinen versio, joka koostui ensimmäisestä versiosta sekä CSS-P:stä ja tämän jälkeen CSS:n versio 2.1 on täydentänyt ja poistanut turhia ominaisuuksia toisesta versiosta. Kolmannen version määritysten tekeminen aloitettiin jo vuonna 1999 ja kehittelytyö kolmannen version parissa on vienyt vuosia. Jotkin uusimmat selaimet tukevat tällä hetkellä joitakin CSS:n kolmannen version ominaisuuksia. (Negrino & Smith 2007, 264; Teague 2004, 8 – 9.)

CSS3 versio tuo tullessaan monia ominaisuuksia, joilla saadaan aikaan entistäkin näyttävämpiä ulkoasuja vähemmällä vaivalla. Esimerkiksi voidaan tehdä varjoja laatikoiden taakse käyttämällä box shadow (laatikko varjo) elementtiä. Pyöristettyjen kulmien tekeminen ennen vaati sen, että jokaiseen kulmaan lisättiin kuva erikseen. CSS3 tarjoaa ominaisuuden, jolla voidaan helposti saada pyöreät kulmat laatikolle. Tekstille on myös CSS3:ssa mahdollista laittaa varjo, jolle määritellään horisontaalinen ja vertikaalinen suunta, sijainti tekstistä sekä varjon väri. (W3schools 2011d.)

Osaksi verkkosivustoa, on olemassa kaksi tapaa ottaa tyylit mukaan, <style>-tagi tai <link>-tagi (Negrino & Smith 2007, 267). Käyttäessä <style>-tagia tulee se lopettaa vastaavaan </style>-tagiin ja se sijoitetaan <head>-tagien sisälle. Esimerkki <style>-tagin käytöstä: <style type=”text/css”>. Kaikki selaimet eivät vaadi tyyppi määritystä, mutta se on kuitenkin hyvä lisätä. Tyyli määritelmät kirjoitetaan <style>-tagien väliin. (Teague 2004, 27 – 28.)

Negrino ja Smith ohjeistaa (2007, 266 – 267), että ulkoisen tyylitiedoston mukaan ottamiseksi on kaksi tapaa. Sen sijaan että määrittelyt kirjoitettaisiin <style>-tagien väliin, voidaan siihen antaa @import-komento ja sen jälkeen polku, mistä tiedosto löytyy. Ulkoisen tyylitiedoston voi myös lisätä osaksi verkkosivua <link>-tagilla. Esimerkki <link>-tagin käytöstä: <link type=”text/css” rel=”stylesheet” href=”tyyli.css”>. Teague selittää (2004, 33) tarkemmin mitä tagin sisälle on kirjoitettu, ensimmäiseksi on tyyppi, toisena kerrottu selaimelle että kyseessä on tyylitiedoston linkitys ja lopuksi annettu polku mistä tiedosto löytyy.

Teague kertoo (2004, 10), että tyylitiedoston sisällä erottelu toisistaan tapahtuu eri selektoreilla joko id-attribuutilla, class-attribuuttilla tai HTML selektorilla. Molemmat Teague (2004, 11) sekä Negrino ja Smith (2007, 266) kertovat, jotta tyylit vaikuttaisi verkkosivuun, on eri elementeille annettava siis id tai class-attribuutti ja että HTML-selektorit vaikuttavat HTML-elementteihin. Teague ohjeistaa (2004, 37), että HTML-selektoria käyttäessä, sen eteen ei tule mitään. Negrino ja Smith muistuttavat (2007, 268), että class-attribuutti aloitetaan tyylitiedostossa pisteellä jonka jälkeen on annettu class-attribuutin nimi. Teague kertoo (2004, 41), että id-attribuutti aloitetaan ristikkomerkillä (#) ja tämän jälkeen tulee id-attribuutille annettu nimi.

Tyylitiedostossa on mahdollista myös tehdä joukolle eri elementtejä samat tyylimääritykset. Selektorit listataan peräkkäin pilkulla (,) toisistaan eroteltuna. Lista voi sisältää kaikilla kolmella selektorilla olevia nimiä, kunhan ne noudattavat omia merkkaussääntöjään. (Teague 2004, 42 – 43.) Oli kyseessä yksittäinen tai joukko, jolle tyylejä halutaan määritellä, kirjoitetaan ne aina aaltosulkujen sisälle selektorin nimen jälkeen (Negrino & Smith 2007, 266).

CSS:ssä on olemassa myös luokka, jolla voidaan tehdä efektejä joillekin selektoreille tätä kutsutaan pseudoluokaksi (pseudo-class). Linkkien kanssa speudoluokkia käytetään useimmiten, niillä voidaan määritellä erikseen mitä linkille tapahtuu eri tilanteissa. Pseudoluokkia käyttäessä tulee ne merkitä oikeaan järjestykseen, koska ne saattavat kumota toinen toisensa väärässä järjestyksessä esiintyessä. (Negrino & Smith 2007, 273 – 274; Teague 2004, 46 – 48.)

Kuten aikaisemmin oli kerrottuna, ei ennen CSS-määrityksiä verkkosivuille pystytty määrittelemään elementeille tarkkoja sijainteja. Negrino ja Smith kertovat (2007, 279), että aikaisemmin sijoittelut tapahtuikin taulukoita käyttämällä. Teague lisääkin (2004, 147) tähän, että taulukoilla sijoittelu hidastaa selainta ja CSS:ää käyttämällä saa tarkemman ulkoasun ja nopeammin latautuvan verkkosivun. Sijoittelu tapoja on olemassa neljä erilaista, joilla voidaan ohjata mihin kohtaan elementti verkkosivuilla halutaan. Nämä sijoittelu tavat on: static, fixed, relative ja absolute. Elementtejä voidaan myös halutessa sijoittaa toinen toisensa päälle. Elementit voidaan sijoittaa seuraavilla ominaisuuksilla ylös (top), alas (bottom), oikealle (right) tai vasemmalle

(left). Kaikki sijoittelu tavat eivät tue edellä mainittuja ominaisuuksia, esimerkiksi sijoittelu tapa static ei toteuta näitä ominaisuuksia. (W3schools 2010c.)

Teague (2004, 150-151) kertoo, että elementit käyttää oletuksellisesti static-sijoittelua. Kun käytössä on static-sijoittelu, tarkoittaa se sitä, että elementit esiintyvät verkkosivuilla toinen toisensa jälkeen. Kun käytetään relative-sijoittelua, voidaan elementille antaa arvoja, joiden mukaan se on tietyssä paikassa sivulla, mutta static-sijoittelun mukaisesti toinen toisensa jälkeen. Absolute-sijoittelu mahdollistaa elementeille tarkan paikan, eli se sijaitsee verkkosivulla juuri siellä mihinkä se on asetettu. Fixed-sijoittelua kaikki selaimet eivät tue, mutta se toimii samankaltaisesti absolute-sijoittelun kanssa. Absolute-sijoittelun ja fixed-sijoittelun ero on siinä kun verkkosivustoa rullataan alaspäin, ei fixed-sijoiteltut elementit liiku paikastaan mihinkään.

Kuvassa 2 on havainnollistettu niitä asioita, joita tässä luvussa on käyty läpi.

```
<head>
<link rel="stylesheet" type="text/css" href="tyyli.css">

<style type="text/css">
h3 {
color:red;
text-align:left;
}

#para1 {
text-align:center;
color:red;
}

.para2 {
text-align:center;
color:blue;
position:fixed;
top:30px;
}

h1, h2, .para3 {
color: yellow;
}

a {
color: #000000;
}

a:link {
color: #FF0000;
}
</style>
</head>
```

KUVA 2. CSS esimerkki

2.4 JavaScript

JavaScriptin on kehittänyt Brendan Eich vuonna 1995 työskennellessään Netscapella (Asleson & Schutta 2007, 149). Aslesonin ja Schuttan mukaan (2007, 6) verkkosivujen kehittäjille kehitettiin JavaScript, jotta dynaamisesti tageja HTML-kielessä olisi helpompi muokata. Käyttäjille tahdottiin antaa parempi kokemus verkkosivuilla tällä tavalla.

Verkkosivujen kehittäjien parissa JavaScriptin suosio kasvoi ja tämän huomasi myös Microsoft. Microsoft halusikin sitten omaan Internet Exploreriinsa myös skriptausominaisuuden. Microsoft kehitti oman vastaavanlaisen kielen, joka tunnetaan nimellä JScript. (Negrino & Smith 2007, 5.) Microsoftin JScriptin ja Netscapen JavaScriptin välillä oli tietenkin eroavaisuuksia. Microsoft kehitti JScriptin niin, ettei se ollut täysin yhteensopiva JavaScriptin kanssa, sekä niin että se toimi Microsoftin Internet Explorerissa ainoastaan. (Negrino & Smith 2007, 445.)

Koska Netscapen ja Microsoftin skriptauskielet erosivat toisistaan, kehitettiin standardit kieltä varten. ECMA, joka on kansainvälinen standardointi organisaatio, sai molemmilta yrityksiltä määrityksiä, kommentteja ja ehdotuksia yhteisen standardin luomista varten. ECMA 262 tai ECMAScript nimillä tunnettu standardi aloitettiin vuonna 1996 ja se valmistui vuonna 1997 kesäkuussa. (Negrino & Smith 2007, 447; W3schools 2010a.)

Kuten Negrino ja Smith (2007, 3) toteavat, että vaikka nimi kuulostaa samalta, ei Javalla ja JavaScriptillä ole mitään yhteistä toinen toisensa kanssa. Asleson ja Schutta kertovat (2007, 6), että JavaScriptillä on ollut monta nimeä, ennen kuin se sai lopullisen nimensä. Aluksi, kun JavaScriptiä alettiin kehittämään, kutsuttiin sitä Mochaksi, josta se vaihtui sitten LiveWireksi ja tämän jälkeen vielä LiveScriptiksi. Samalla, kun Netscape julkaisi version kaksi selaimestaan, vaihtui nimi JavaScriptiksi (Negrino & Smith 2007, 5).

Negrino ja Smith (2007, 5) selittävät nimen muuttuneen sen takia, että Netscape toivoi Sunin luoman Javan ja sen suuren huomion kautta osan loistosta tarttuvan myös

JavaScriptiin. Tämä oli siis markkinointi kikka, josta myöskin Asleson ja Schutta (2007, 149) sanovat lopullisen nimen johtuneen.

Jotta JavaScript saataisiin mukaan osaksi verkkosivua, on olemassa kaksi tapaa liittää skripti ohjelmaan. Aina JavaScriptiä mukaan liittäessä on käytettävä <script>-tageja sekä lopetettava ne vastaavan lopetustagiin, </script>-tagi lopettaa skriptin. Skripti on siis mahdollista kirjoittaa suoraan tagien väliin tai tehdä ulkoinen oma tiedosto skriptiä varten. Ulkoisessa skriptissä ei saa olla aloitus <script> ja lopetus </script>-tageja. (Negrino & Smith 2007, 2; W3schools 2010a.)

Skriptin voi HTML-dokumentissa sijoittaa joko <head> ja </head>-tagien väliin tai sitten vaihto ehtoisesti <body> ja </body>-tagien väliin. Molempiin osioihin voidaan laittaa niin monta skriptiä, kun halutaan, sekä skriptejä voi sijoittaa molempiin osioihin saman aikaisesti. (W3schools 2010a.)

Ennen kuin selain ymmärtäisi mikä skriptauskieli on kyseessä on lisättävä tagin sisään language=”Javascript”-attribuutti. Myöskin selaimelle on kerrottava, että teksti koostuu JavaScripti-koodista, se tapahtuu lisäämällä attribuutin type=”text/javascript”. Ulkoisen skriptin lisääminen osaksi sivua tapahtuu src-attribuutilla, lisäämällä siihen skripti tiedoston nimen ja mahdollisen polun, esimerkiksi seuraavalla tavalla: src=”javascript.js”. Pelkkää JavaScriptiä sisältävä tiedosto tulee loppua päätteeseen .js. Tagin lopetuksen yhteydessä ei tarvitse lisätä mitään attribuutteja. (Negrino & Smith 2007, 23 – 26.) Kuvassa 3 on esitelty tavat joilla skriptin saa osaksi ohjelmaa.

```
<html>
<head>

<script language="Javascript" type="text/javascript" src="javascript.js"></script>

<script language="Javascript" type="text/javascript">
        document.write("Hello World!");
</script>

</head>
<body>

<script language="Javascript" type="text/javascript">
        document.write("Hello World!");
</script>

</body>
</html>
```

KUVA 3. Skripti esimerkki

JavaScriptillä voidaan vaikuttaa sivuston eri elementteihin, kuten kuviin, linkkeihin tai mihin tahansa muuhun elementtiin, käyttämällä class (luokka) ja id-attribuuttia. Class-attribuuttia käytetään silloin, kun halutaan vaikuttaa useammin kuin yhden kerran ja id-attribuuttia enemmänkin silloin, kun jokin elementti esiintyy vain kerran sivuilla. JavaScriptin lisäksi CSS käyttää näitä attribuutteja ja kun elementeille on annettu id tai class-attribuutti, voidaan elementtien toimintaa ja ulkoasua muokata molemmilla kielillä. (Negrino & Smith 2007, 18 – 19.)

Välttääkseen sivustoa suorittamasta skriptejä sivun vielä latautuessa, voidaan skriptit kirjoittaa funktioiden sisälle. Nämä funktioiden sisällä olevat skriptit suoritetaan vain kutsuttaessa. Jotta oltaisiin varmoja siitä, että funktio on varmasti valmiina toimintaan, olisi funktiot hyvä kirjoittaa HTML-dokumentin <head>-osioon. (W3schools 2010a.) Negrino ja Smith kertovat (2007, 25), että jokaiselle funktiolle on annettava nimi sekä funktioita voidaan halutessa kutsua myös muista skriptin osista.

Tapahtumat ovat sellaisia toimintoja, jotka JavaScript huomaa. Sivuston jokaisella elementillä, on jokin mahdollinen tapahtuma, joka voi laukaista JavaScriptin. Hiiren liike, hiiren painikkeen painaminen, näppäimistön näppäimen painaminen tai esimerkiksi jonkin lomakkeen lähettäminen voi sisältää tapahtuman, joka laukaisee funktion toiminnan. (W3schools 2010a.)

2.5 jQuery

jQueryn kehittäjä John Resig aloitti vuonna 2005 aluksi tekemään JavaScript kirjastoa CSS valitsimille, joka oli syntaksiltaan ytimekkäämpi kuin muut olemassa olevat kirjastot. Vuonna 2006 jQuery projekti ilmoitettiin ensimmäistä kertaa BarCampNYC sivustolla ja myöhemmin samana vuonna jQuery 1.0 julkaistiin. jQueryn kehitys jatkui ja tammikuussa 2010 julkaistiin jQuery 1.4. (jQuery Project 2011.)

John Resig artikkelissaan jQuery, Nokia ja Microsoft kertoo, että syyskuussa 2008 Nokia ja Microsoft ovat alkaneet tukemaan jQueryä. Microsoft halusi ottaa jQueryn osaksi heidän virallista kehitysalustaansa. Nokia halusi käyttää jQueryä kehittämään sovelluksiaan heidän WebKit-pohjaiseen Web Run Timeen. Nokia halusikin aloittaa ensimmäisenä siirtämään lukuisia sovelluksiaan käyttämään jQueryä. jQuery projektin kehitys joukko oli iloinen saadessaan kaksi suurta yhtiötä tukemaan heitä. Lisäksi jQuery projekti pystyy saamaan kehitysehdotuksia ja testaus tuloksia molemmilta yhtiöiltä, jotka analysoidaan ja tarkastetaan jonka jälkeen ne hyväksytään tai hylätään. (Resig, John 2008.)

jQuery on tiivis JavaScript kirjasto, jonka avulla saadaan aikaan yksinkertaisempia HTML-dokumentteja, voidaan tehdä animaatioita sekä tapahtuma käsittelijöitä ja vaikuttaa Ajaxiin (jQuery 2011). Kuka tahansa voi ladata jQuery kirjaston osaksi verkkosivuaan. jQuery kirjaston lisääminen tapahtuu normaalilla skriptin lisäyksellä: <script type="text/javascript" src="jquery.js"></script>. (Resig, John 2011.)

jQueryn syntaksi on räätälöity valitsemaan HTML elementtejä ja suorittamaan jonkin tapahtuman elementille. Perus syntaksi näyttää tältä: $(selector).action(). Dollarin ($) merkki alussa määrää, että se on jQueryä ja se on lyhenne sanalle jQuery. Muissakin JavaScript kirjastoissa käytetään samaa tapaa merkitä funktiot, siksi jQuery tarjoaakin noConflict()-metodin, jolla voidaan antaa uniikki nimi osoittamaan jQueryä, esimerkiksi jq. Sulkeisissa olevan selector sanan tilalle laitetaan HTML-elementti, johon halutaan vaikuttaa. Lopuksi action() komennolla kerrotaan, mitä elementille halutaan tapahtuvan. (W3schools 2011f.)

Efektejä elementeille voidaan jQuery kirjaston avulla tehdä. Erilaisia efektejä on esimerkiksi piilota, näytä, liukuminen esiin tai piiloon, näyttäminen tai katoaminen hitaasti. Esimerkiksi näytä ja piilota eli show() ja hide() funktiot saavat kaksi vaihtoehtoista parametria, speed(nopeus) ja callback. Speed-parametrille voidaan määritellä onko se nopea(fast), normaali(normal) vai hidas(slow) tai aika millisekunneissa. Funktio nimeltä callback käynnistetään vasta sen jälkeen, kun animaatio (efekti) on loppunut. JavaScriptissä komennot suoritetaan rivi riviltä, kuitenkin animaatioiden kanssa seuraava rivi voidaan käynnistää vielä animaation ollessa kesken ja tämä aiheuttaa virheitä. Virheiden estämisen vuoksi luodaan callback-funktio, joka kutsutaan vasta animaation loputtua. (W3schools 2011f.)

2.6 Ajax

Ajax termi koostuu sanoista Asynchronous JavaScript and XML, verkkosovellusten kehittäjä sekä kirjoittaja Jesse James Garrett keksi tämän termin vuonna 2005. Ajax koostuu monesta eri teknologiasta kuten XHTML:stä, CSS:stä, DOM-mallista, XML:stä ja XMLHttpRequestista. (Negrino & Smith 2007, 9.) Asleson ja Schutta (2007, 14) mainitsevat, ettei Ajax ole mikään uusi juttu, koska se koostuu kaikesta jo ennestään tutuista tekniikoista. XMLHttpRequest on näistä uusin tekniikka, joka julkaistiin vuonna 1999 osana viidettä versiota Internet Explorer-selainta ActiveX:n komponenttina. Aluksi Internet Explorer oli ainut selain joka tuki tätä komponenttia, mutta myöhemmin Mozilla ja Safari alkoi tukea sitä.

Ajax-tekniikalla saadaan aikaan sellaisia sovelluksia, että ne vastaavat käyttäjän komentoihin heti, eikä käyttäjän tarvitse odotella sivun päivitystä. Ajaxilla halutaan saada aikaan työpöytäsovelluksia muistuttavia käyttöliittymiä. (Negrino & Smith 2007, 357.) Asleson ja Schutta (2007, 14 – 15) kertovat millaisia dynaamisesti toimivia sovelluksia Ajax-tekniikalla voidaan saada aikaan. Esimerkiksi liukusäätimiä, missä käyttäjä voi liikuttaa säädintä haluamaansa kohtaan ja sivu päivittyy dynaamisesti koko ajan eikä vasta, kun jokin toiminto on tehty. Voidaan myös tehdä sellaisia sovelluksia, joissa tulee niin sanottuja infokuplia esiin, kun hiiren kursori menee jonkin elementin päälle, kuten esimerkiksi kuvan.

Molemmat Asleson ja Schutta (2007, 14 – 15) sekä Negrino ja Smith (2007, 10) mainitsevat Googlen käyttävän Ajaxia paljonkin hyödykseen, koska sillä saadaan aikaan dynaamisesti toimivia verkkosivuja ilman mitään lisälaitteita tai plug-in -ohjelmia. Toisena yrityksenä Yahoo! on luonut paljon verkkosovelluksia Ajaxilla ja täten kuuluu Ajaxia tukevaan joukkoon. Kuka tahansa voi käyttää sekä luoda uusia omia verkkosovelluksia pohjautuen Googlen tai Yahoo!:n verkkosovelluksiin, tämä johtuu siitä, että molemmat yritykset ovat tehneet rajapinnoistaan julkisia.

XMLHttpRequest oli aluksi vain osana Internet Explorer-selaimen viidettä versiota ActiveX-komponenttina. Myöhemmin myös Mozilla sekä Safari-selaimet alkoivat tukea tätä tekniikkaa. Ilman W3C:n standardia oli XMLHttpRequestin toiminnallisuus eri selaimissa vaihtelevaa. (Asleson & Schutta 2007, 25.) Ensimmäinen määritelmä standardia varten tehtiin vuonna 2006. Standardin määrittelyiden takana on W3C:n verkkosovellusten työryhmä. (Jackson & Kesteren 2006.) Uusin määrittely XMLHttpRequestista on julkaistu vuonna 2010, jossa mainitaan että aikaisintaan helmikuun alussa vuonna 2011 on tiettyjen kriteerien täytyttävä, jotta XMLHttpRequest pääsisi määrittely ehdokkuudesta pois (Kesteren 2010a).

XMLHttpRequest on oikea unelma ohjelmoijalle neljästä syystä. Ensinäkin sen avulla voi päivittää verkkosivuja, ilman että sivua tarvitsisi uudelleen ladata, toiseksi ja kolmanneksi syyksi siksi, että sillä voidaan lähettää ja vastaan ottamaan tietoa serveriltä, kun verkkosivu on latautunut. Neljännes syy on se, että tietoa voidaan lähettää serverille taustalla. (W3schools 2011c.)

Selaimen ja serverin välinen toiminta Ajax-sovelluksessa alkaa, kun tapahtuma esiintyy. Aluksi luodaan XMLHttpRequest-objekti ja se lähetetään serverille. Serverillä HttpRequest käsitellään ja sen jälkeen luodaan vastaus ja lähetetään takaisin selaimelle. Lopuksi käydään takaisin lähetetty tieto JavaScriptillä läpi ja suoritetaan mahdollinen verkkosivujen päivitys. (W3Schools 2011e.)

Asleson ja Schutta (2007, 27 – 28) kertovat tavanomaisimpia XMLHttpRequest-metodeja olevan esimerkiksi open() ja send(). Yhteyden avaaminen tapahtuu open-metodilla, joka vaatii kaksi pakollista parametria ja kolme valinnaista. Pakollisina parametreina ovat metodi ja url (osoite), eli lähetysmuoto joko GET, POST tai PUT ja

osoite minne pyyntö lähetetään. Pyyntö palvelimelle lähetetään send-metodilla, valintaisena parametrina annetaan esimerkiksi merkkijonotyyppistä tekstiä.

Lähetysmuodoissa POST ja GET eroavaisuuksina on se, että POST lähettää tiedon pyynnön eli send-metodin osana ja GET lähettää tiedon osana yhteyden avaamista eli open-metodin url-parametrin osana. Tiedon tallentamiseen ja päivittämiseen suositellaan käyttämään POST-metodia ja GET-metodia vain silloin kun tietoa otetaan vastaan. (Asleson & Schutta 2007, 58.) POST-metodilla pystytään lähettämään suuria määriä tietoa serverille ja POST-metodi on myös vakaampi ja turvallisempi tapa kuin GET-metodi (W3schools 2011e).

2.7 Adobe Flash

Flashin historia alkaa siitä, kun FutureWave niminen yritys julkaisi vuonna 1996 tuotteen nimeltä FutureSplash Animator (Asleson & Schutta 2007, 9). Paananen (2008, 7) kertoo, että Macromedia-ohjelmisto taloa on pidetty Flash-formaatin keksijänä, mutta FutureWave sulautti itsensä Macromedia yritykseen vuonna 1996 ja näin Macromedia sai tämän tekniikan. Asleson ja Schutta jatkavat (2007, 9), että tämän jälkeen tuote uudelleen nimettiin Flashiksi.

Macromedia halusi päästä versionumeroinnista eroon ja Flash 5.0 jälkeen vuonna 2002 julkaistiin Flash MX. Tämä versio korjasi vakausongelmia, joita aikaisemmissa versioissa oli esiintynyt, sekä Flash MX:n käyttöliittymä oli päivitetty muiden Macromedian ohjelmien kanssa yhtenevään suuntaan. Vuonna 2004 julkaistiin Flash MX2004, joka jäi sarjan viimeiseksi ja tämän jälkeen alettiin käyttämään taas versionumeroita. Vuonna 2005 julkaistiin Flash 8, joka oli Macromedian viimeisin julkaistu Flash. Yritys nimeltä Adobe osti Macromedian ja vuonna 2007 osana Creative Suite CS3-pakettiaan he julkaisivat ensimmäisen Adobe Flashinsä. (Paananen 2008, 7.)

Flash on helppoa eikä vaadi ohjelmointitaitoja ja sillä saadaan aikaan dynaamisia sovelluksia. Toimiakseen Flash kuitenkin vaatii sen, että käyttäjä on asentanut ohjelmiston. Monet käyttöjärjestelmät sekä selaimet sisältävät jo Shockwave Player plug-in -osan eli liitännäisohjelman, jolla mahdollistetaan Flashin toimivuus. (Asleson

& Schutta 2007, 9 – 10.) Korpela ja Linjama (2005, 261) kertovat, että Flashin esittämistä verkkosivuilla varten tarvittava ohjelma on myös saatavilla ilmaiseksi verkossa.

Paananen (2008, 7) kertoo yleisimpiä käyttömuotoja Flashille sen lisäksi, että sillä voidaan tehdä verkkosivuja. Melkeinpä alusta asti Flashiä on käytetty verkkosivuilla bannerimainontaan, koska Flash-tekniikalla voidaan pieneen tiedostokokoon laittaa erilaisia ominaisuuksia, joilla voidaan aktivoida käyttäjää. Flash toimii myös video alustana ja palvelut kuten YouTube on käyttänyt sitä. Flashillä pystytään tekemään monipuolisia asioita ja jopa tekemään presentaatioita, tämä korvaa esimerkiksi Microsoftin Power Point-ohjelman.

Paananen kertoo (2008, 7), että Flash on yksi maailman käytetyin multimediaformaatti. Adoben verkkosivuilla julkaistussa tutkimuksessa joulukuussa vuonna 2010 Euroopassa noin 99.5 prosenttia oli asentanut Flash Player 10 version. Tutkimuksessa käytetyt Euroopan maat olivat Ranska, Saksa ja yhdistyneet kuningaskunnat. (Adobe 2011.)

Kuvilla, teksteillä, äänillä, videoilla ja komennoilla saadaan aikaa Flash-animaatio (Paananen 2008, 7). Flash-animaatio koostuu aikajanasta, jonka kulkuun käyttäjä voi vaikuttaa. Flashin rakentamiseen voidaan käyttää yhtä tai useampaa tasoa. Tasot koostuvat avainkehyksistä sekä avainkehyksien väliin jäävistä kehyksistä. Tasoja sekä kehyksiä voi luoda rajoittamattomasti ja Flash-animaation pituus riippuu siitä kuinka monta kehystä on. (Paananen 2008, 12.)

Asleson ja Schutta (2007, 9) totesivat, että Flash on helppoa koska sen tekeminen ei vaadi ohjelmointitaitoja. Paananen (2008, 19) esittelee ActionScript-ohjelmointikielen, jolla voidaan tehdä toimintoja Flash-animaatioon. Yksinkertaisimpiä komentoja ovat esimerkiksi aikajanan pysäyttäminen ja moni mutkikkaammilla komentosarjoilla voidaan aikaan saada jopa animaatioita ja grafiikkaa. ActionScript versioitakin on useampia, joista ActionScript 3.0 on uusin, sen mukana tuli paljon uudistuksia ohjelmointikieleen.

Paananen kertoo (2008, 21) muutamia yleisimpiä ActionScript komentoja, koska niitä on olemassa tuhansia, mutta niitä kaikkia ei kuitenkaan perustyöskentelyssä tarvita. Flash-animaation etenemistä voidaan kontrolloida toista (Play) ja pysäytä (Stop) komennoilla. Flash-animaatio etenee ruutu kerrallaan, joten pysäyttäminen voidaan sijoittaa johonkin avainkehykseen tai pysäytys komento voidaan sijoittaa painikkeeseen, jotta käyttäjä voi painiketta painamalla pysäyttää animaation. Toista (Play) komento on yleensä painikkeena ja tällä saadaan animaation jatkuminen aikaan. On olemassa myös goto-komento eli siirtyminen, jolla voidaan hypätä haluttuun kohtaan animaatiossa. Tämä komento voidaan sijoittaa painikkeeseen tai kehykseen ja se siirtyy siitä haluttuun kohtaan aikajanalla.

2.8 Kilpailevat tekniikat Microsoft Silverlight ja JavaFX

Kehittelyalustoja dynaamisia verkkosivuja varten Flashin ohella on olemassa myös Silverlight ja JavaFX. Moroney (2008, xiv) kertoo, että Silverlight on Microsoftin kehittämä liitännäisohjelma selaimia varten. JavaFX on nykyisin Oraclen omistama liitännäisohjelma selaimia varten, jonka kehitti Chris Oliver (Oliver 2006). Molemmilla JavaFX:llä ja Silverlightilla voidaan tehdä työpöytäsovelluksia sekä ohjelmia selaimille, puhelimille ja kodinkoneille (Moroney 2008, 3 – 4; Oracle Corporation 2010).

Moroney kertoo (2008, 3), että Silverlight on selain ja kehitysalusta riippumaton liitännäisohjelma, jonka tarkoitus on luoda parempi kokemus käyttäjälle. Silverlight versiota 1.0 pystyi ohjelmoimaan sellaisilla tekniikoilla kuten Ajaxilla, JavaScriptillä ja DHTML:llä. Silverlight versio 2.0 lisäsi dynaamisuutta ja .NET kieli tuen, sekä uusia ominaisuuksia joita on mahdollista käyttää ainoastaan .NET kehitysalustalla.

Silverlight käyttää XAML:ia helpottaakseen käyttöliittymä kehitystä, näitä on esimerkiksi animaatiot, grafiikka ja sommittelu. Silverlight toimii kaikissa suurimmissa selaimissa ja sillä voidaan katsella videoita ja kuunnella ääntä. Silverlight mahdollistaa käyttäjälle raahaa ja pudota, sekä suurennus ominaisuudet selaimella. Tällä hetkellä uusin julkaistu Silverlight on versio numero 4. (Microsoft Corporation 2011.)

JavaFX:stä on julkaistu tällä hetkellä versio numero 1.3.1 ja JavaFX 2.0 on kehitteillä. JavaFX 2.0 esimerkiksi saa uuden media ominaisuuden, jolla pystytään katselemaan videoita koko ruudulla. Myöskin voidaan määritellä kohta videolle mihinkä asti se lataa eli streamaa itseään tämä mahdollistaa sen, että näihin kohtiin voidaan asettaa jokin tapahtuma, esimerkiksi mainos. Myös JavaFX pystyy siihen mihin Silverlight, kuten grafiikan, videoiden, äänien ja animaation näyttämiseen ja tekemiseen. JavaFX ohjelmat kirjoitetaan JavaFX Scriptiin, joka on yksinkertaista ja helppo oppia. (Oracle Corporation 2010.)

3 KORTTILAJITTELU

Psykologit kehittivät korttilajittelun, jotta he voisivat tutkia kuinka ihmiset organisoivat ja kategorisoivat omia tuntemuksiaan. Näille koehenkilöille näytettiin kortteja, joissa oli joko jotain konkreettista tai abstraktia. Kortit piti lajitella pinoihin niin, että samassa pinossa oli samaa asiaa muistuttavia ja sen jälkeen antaa pinolle jokin nimi. (Wood & Wood 2008, 1.)

Wood ja Wood (2008, 1) kertovat, että kehittäjät jotka tekevät työpöytä- tai verkkosovelluksia, kohtaavat ongelmia kuinka järjestää osiot, ominaisuudet ja toiminnot niin, että käyttäjä löytäisi ne helposti. Korttilajittelu voi olla hyvinkin tehokas tapa löytää paras tapa organisoida informaatioarkkitehtuuri mahdollisten käyttäjien näkökulman mukaisesti. Myöskin Robertson (2001, 1) toteaa, että on haasteellista organisoida informaatioarkkitehtuuri niin, että se olisi käytännöllinen ja mielekäs käyttäjille. Huolellisella tutkimisella ja analysoinnilla informaatiota voidaan saada vihjeitä siitä, mitkä aiheet tulisi olla samassa ryhmässä. Korttilajittelulla ei saada suoraan valmista rakennetta, mutta se antaa kuvan siitä mitenkä käyttäjät näkevät sen.

3.1 Toteutus ja menetelmät

Ensimmäinen askel korttilajittelussa on tehdä lista aiheista, joita testissä tullaan käyttämään. Aiheita ei saa olla liian vähän, koska näin tulee liian pieni näkyvyysalue koehenkilöille tehdä rakenne. Aiheita ei saa olla liian monta, jotta testistä ei tulisi liian

raskas ja epäselvä koehenkilölle. Ryhmä muotoisten termien, ”manuaali” ja ”ohjeet” tapaisia sanoja tulisi välttää, kun aiheita kirjataan korteille. (Robertson 2001, 3.) Wood ja Wood mainitsevat (2008, 4), että aiheita voidaan myös pilkkoa koehenkilöiden kesken siten, että osalle koehenkilöitä annetaan osa aiheita ja osalle koehenkilöitä annetaan toinen osa aiheita, jotka pitää kategorisoida.

Robertson (2001, 6) kehottaa valitsemaan koehenkilöitä, jotka ovat mahdollisia järjestelmän käyttäjiä. Ei ole mitään järkeä antaa esimerkiksi johtajien suorittaa testiä elleivät he ole kohderyhmä ja järjestelmän käyttäjiä. Wood ja Wood (2008, 2) ehdottavat, että 25 – 30 koehenkilöä on riittävä määrä korttilajittelu testin tekemiseen. Nielsen (2004) taas sanoo, että riittävä määrä koehenkilöitä on 15 ja suuremmissa projekteissa 30, sillä enempi määrä koehenkilöitä ei vaikuta suuresti tulokseen. Nielsen sanoo myös ettei suosittele käyttämään vain viittä koehenkilöä, kuten monissa käytettävyys testeissä käytetään, koska näin ei saada tarpeeksi hyviä tuloksia. Wood ja Wood sekä Nielsen yhteisesti toteavat sen, että liian suuri määrä koehenkilöitä nielee liikaa aikaa ja rahaa.

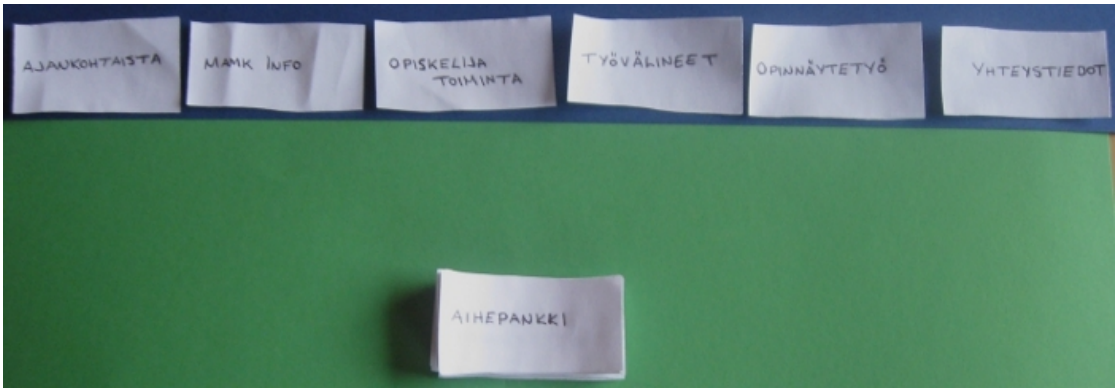
Wood ja Wood (2008, 2 – 3) kertovat, että on olemassa kaksi tapaa toteuttaa korttilajittelu, avoin ja suljettu menetelmä. Avoimessa menetelmässä korteille on valmiiksi kirjoitettu aiheet ja koehenkilön tulee vain kategorisoida ne oman näkemyksensä mukaisesti. Tässä menetelmässä voidaan kategorisoida niin ylä- kuin alatason aiheet. Suljettua menetelmää on hyvä käyttää, silloin kun rakenne on jo olemassa, mutta se kaipaa hiomista. Tässä menetelmässä voidaan myös antaa mahdollisuus uudelleen nimetä aiheita. Wood ja Wood suosittelevat käyttämään avointa menetelmää, koska suljetussa menetelmässä aiheiden uudelleen nimeämisessä voi tulla ongelmia. Nielsen (2004) sanoo, että onneksi nämä kaksi tapaa voidaan myös yhdistää.

Robertson (2001, 7) suosittelee, että lajittelun tapahtuessa paikalla tulisi olla joku henkilö joka tuntee sisällön jo ennestään. Sisällön tunteva henkilö voi vastata koehenkilölle, jos tämä ei ymmärrä täysin aiheen sisältöä. Ennen kuin korttienlajittelu voidaan aloittaa, täytyy koehenkilöitä ohjeistaa selvästi mitenkä se tapahtuu. Koehenkilöille tulee myös kertoa, että heidän tulee organisoida aiheet sillä tavalla, että ne vastaavat heidän tarpeitaan. Robertson ohjeistaa myös ottamaan kynän ja paperia

mukaan korttilajittelun tapahtuessa, jotta koehenkilön kommentit voidaan kirjata. Nielsen (2004) on samaa mieltä, että kommentit on hyvä kirjata ja sanookin, että kaikkein tärkein tieto saadaan koehenkilöiltä heidän kommenteistaan korttienlajittelun tapahtuessa. Näin saadaan tietoon, miksi koehenkilö sijoittaa jonkin aiheen tiettyyn kasaan tai tietyn aiheen alle.

Wood ja Wood (2008, 5) pohtivat pitäisikö koehenkilölle antaa aikaa miettiä ja pohtia aiheita vai pitäisikö aiheet kategorisoida hieman vauhdikkaammin. He tulevat siihen tulokseen, että kun kyse on internetistä, ihmiset eivät jää miettimään ja pohtimaan, että onko jokin linkki se mistä he haluamansa tiedon löytävät. Woodin ja Woodin tutkimusten mukaan, koehenkilöt joita kehotettiin tekemään päätökset hieman vauhdikkaammin ja koehenkilöt jotka saivat aikaa miettiä, lajittelu tulokset eivät eronneet huomattavasti toisistaan. Robertson (2001, 5) kehottaa, että koehenkilön tulisi täyttää lomake, jossa kysytään koehenkilön työtehtäviä ja kauanko on ollut työtehtävissään, mitkä ovat koehenkilön tavat löytää tietoa ja mitä tietoja koehenkilö useimmiten käyttää, mitä muuta tietoa koehenkilö toivoisi löytävän verkkosivuilta sekä muita kommentteja ja ehdotuksia.

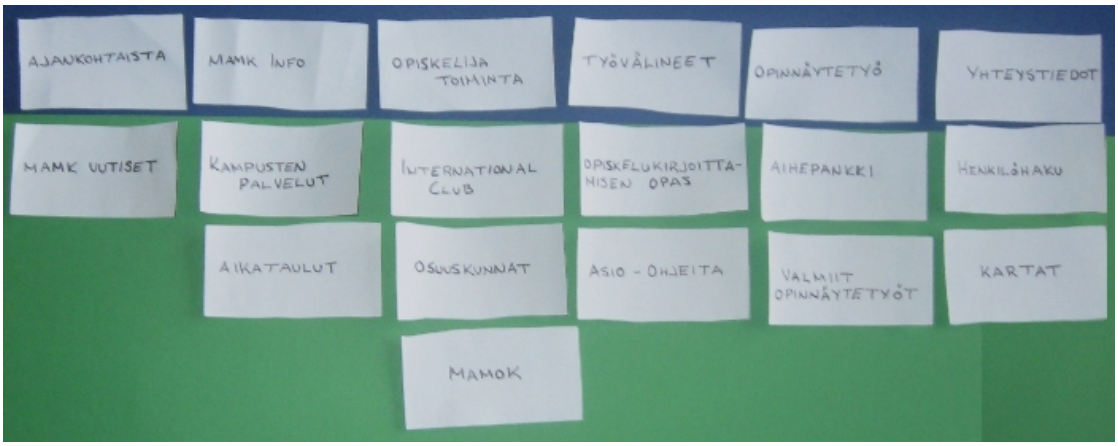
Kuvassa 4 on esiteltynä korttilajittelun lähtötilanne. Ylätason aiheet on valmiiksi paikoilleen aseteltu sinisellä alueella ja alatason aiheet on pinossa vihreällä alueella.



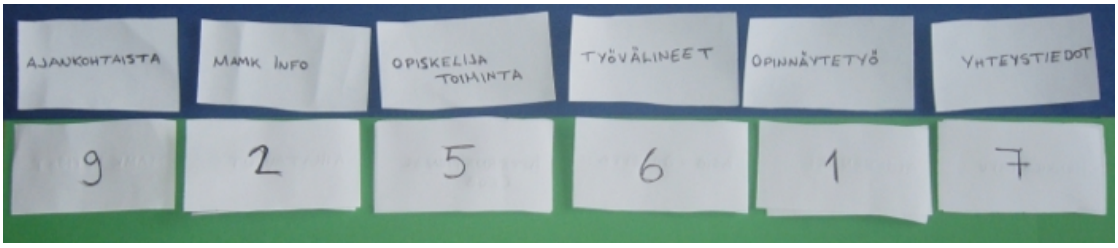
KUVA 4. Esimerkki korttilajittelun lähtötilanteesta

3.2 Tulokset ja analysointi

Robertson (2001, 9) ohjeistaa, että korttilajittelun päätyttyä koehenkilöitä kiitetään. Pirinen (2009, 21) sanoo, että tulokset kirjataan paperille, josta ne voidaan siirtää myöhemmin esimerkiksi Excel-taulukkoon ja varmistukseksi tulokset voidaan myös valokuvata kuvan 5 esimerkin mukaisesti. Robertson (2001, 9 – 10) sanoo, että tulosten keruun helpottamiseksi korteille tulisi kirjata numero, jotka on esimerkiksi merkitty kortin kääntöpuolelle kuten kuvassa 6 näytetään. Robertson jatkaa, että on monta tapaa kirjata tulokset ja itse esittelee muodon, jossa tulokset on viety tauluihin Excel-taulukon sijaan.



KUVA 5. Esimerkki korttilajittelun tuloksista



KUVA 6. Esimerkki korttien numeroinnista

Pirinen (2009, 24) kertoo, että antoi testissään koehenkilöiden pisteyttää ylätasot siinä järjestyksessä, missä koehenkilöt kokivat niiden olevan tärkeysjärjestyksessä. Lopuksi hän siirsi tulokset Excel-taulukkoon ja laski jokaisen ylätason keskiarvon. Ylätasojen keskiarvon perusteella saatiin muodostettua ylätason järjestys. Myöskin Robertson (2001, 5) esittelee pisteytys järjestelmän, jossa koehenkilöt antoivat pisteitä sen mukaan kuinka usein käyttävät tiettyä aihetta.

Alatasojen paikka ja järjestys lasketaan myös Excel-taulukossa. Robertson (2001, 10) kertoo, että tulosten analysointi vaiheessa, jos huomataan monen koehenkilön tehneen samankaltaisen rakenteen, voidaan luottaa siihen, että se on oikeanlainen. Suurien eroavaisuuksien kohdalla, Robertson suosittelee tutkimaan mistä se johtuu. Joka tapauksessa molemmat lopputulokset tulee ottaa huomioon rakenteen suunnittelussa.

Pirisen (2009, 24 – 25) tekemän testin, mukaan alatasot voivat sijoittua monenkin ylätason alle. Tämä johtuu siitä, kun alatasolla on tarpeeksi suuri prosentti monessa ylätasossa. Riippuen menetelmästä ja kuten Pirinen sanoo, että koehenkilöt saivat käyttää ”tuplakortteja”, voi alatasojen prosentti määrä kasvaa tietyn ylätason alla.

4 SOVELLUKSEN TOTEUTUS

Esimerkki toteutus on sovellus, joka toimii selaimella. Sovelluksen tarkoituksena on helpottaa navigaatiorakenteen tekemistä korttilajittelu tekniikalla ja helpottaa sekä nopeuttaa tulosten tallentamista. Sovelluksella toimiva korttilajittelu säästää aikaa ja rahaa sekä on ekologinen, kun kortteja ei tarvitse tehdä paperille tai kartongille. Tulokset voidaan hakea sovelluksella Excel-taulukkoon projekti kohtaisesti tietokannasta. Esimerkki toteutus on tehty php:llä, HTML5:llä, JavaScriptillä sekä siinä on käytetty Ajax-tekniikkaa tiedon tallentamiseen tietokantaan, tietokantana on käytetty MySQL:ää. Tiedon tallennukseen olisi voinut käyttää myös jQueryä, mutta päätin käyttää Ajax-tekniikka, koska se oli minulle tutumpi. Sovelluksen toimivuus on parhaaksi havaittu Mozilla Firefox:lla ja Google Choremella, koska nämä kaksi selainta tukevat tällä hetkellä HTML5:tä parhaiten.

Opinnäytetyön tekeminen alkoi siitä, että ensin minun piti tutustua HTML5:den tarjoamaan drag and drop (vedä ja pudota) tekniikkaan. Tutustumisen jälkeen aloin tekemään esimerkki sovellusta. Esimerkki sovelluksessa on käytetty Mikkelin ammattikorkeakoulun opiskelija intranetin (studentin) navigaatiorakennetta.

Sovelluksessa kortit koostuvat listaelementeistä, joille on annettu draggable-attribuutti. Tapahtuma attribuuteilla ohjataan, sitä mitä elementille tapahtuu. Elementillä pitää olla id, jotta se voidaan tunnistaa ja sovellus tietäisi mitä elementtiä,

sen tulee käsitellä. Kuvassa 7 on esimerkki siitä, mitenkä lista elementille annetaan attribuutit, ylempi on ylätason kortti ja alempi on alatason kortti.

```
<ul id='$paataso_id' draggable='true' ondragstart='return dragDefine(event)'
ondragover='return dragOver(event)' ondrop='return dragDrop(event)'> Opinnäytetyö </ul>

<li id='$alataso_id' draggable='true' ondragstart='return dragDefine(event)'
ondragend='dragEnd(event)'> Aihepankki </li>
```

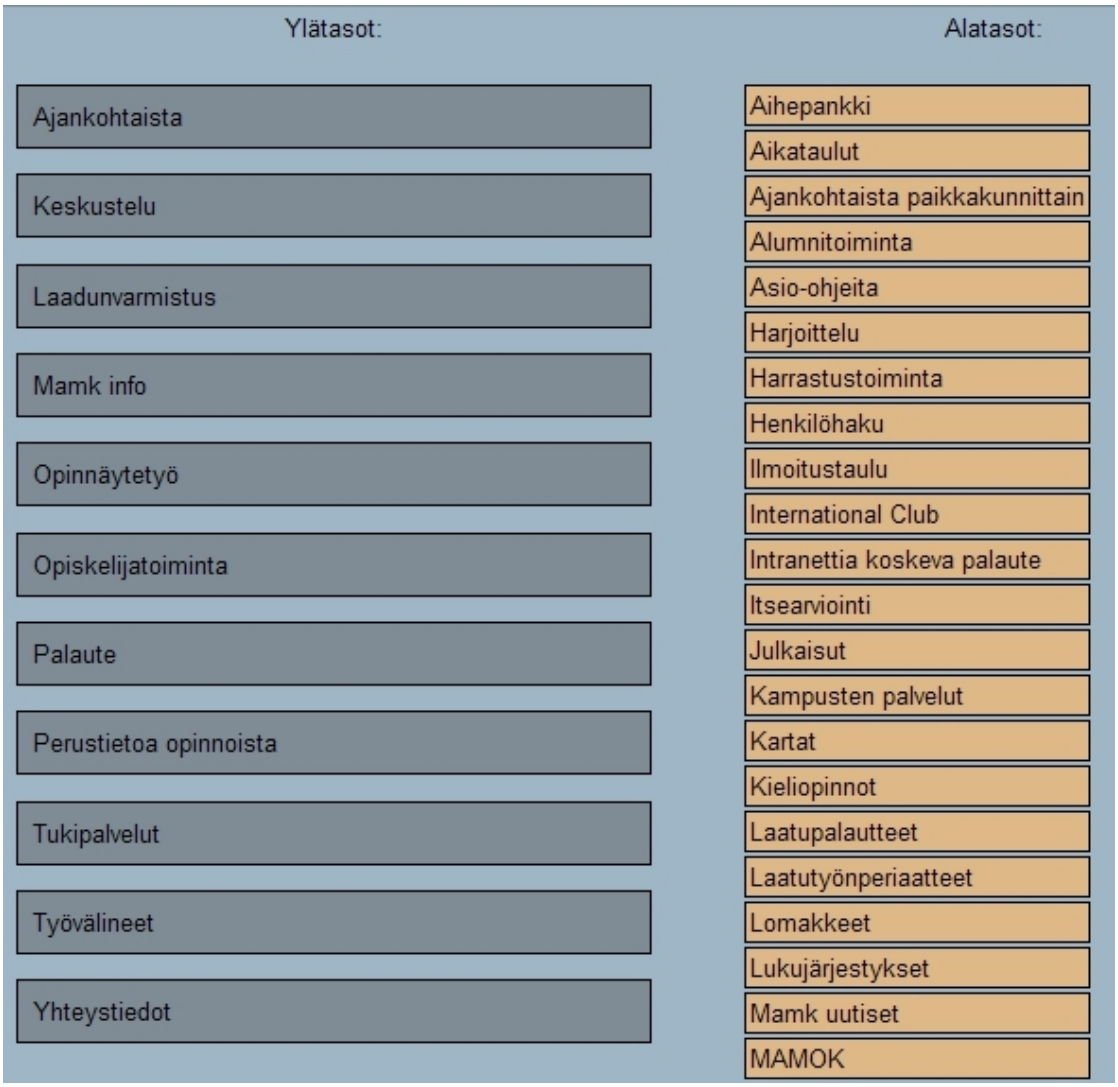
KUVA 7. Esimerkki koodi

Seuraavassa listassa on kerrottu, mitä jokainen attribuutti tekee:

- ondragstart: skripti joka ajetaan, kun drag (vedä) operaatio alkaa
- ondragover: skripti joka ajetaan, kun elementti on alueella, jossa pudotus on mahdollista
- ondrop: skripti joka ajetaan, kun elementti pudotetaan
- ondragend: skripti joka ajetaan drag (vedä) operaation lopussa

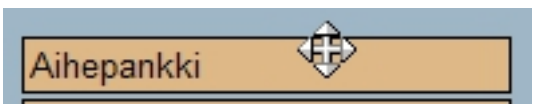
Drag and drop-tekniikan avulla koehenkilö pystyy järjestelemään ylätasot ja alatasot paikoilleen. Koehenkilö aloittaa järjestelemällä ylätasot ja tämän jälkeen siirtää alatasot sen ylätason alle, mihinkä kokevat sen kuuluvan. Kuvassa 8 on esitelty, miltä sovelluksen ulkoasu näyttää. Vasemmalla on ylätason kortit ja oikealla on alatason kortit.

Projektista riippuen voi ylä- ja alatasoja olla enemmän tai vähemmän. Sovellusta tehdessäni törmäsin ongelmaan, jossa ylätasoista tuli niin pitkiä, että alatasoja oli vaikea hakea ja vetää paikoilleen, kun osa alatasoja oli jo paikoillaan. Korjasin tämän sillä lailla, että alatasot ovat listana, siten etteivät kaikki näy kerralla ja loput jäävät piiloon. Ylätasoista voi tulla niin pitkiä, että koehenkilö joutuu rullaamaan sivua alemmas, tällöin alatasot liikkuvat mukana ja koehenkilön on helpompi siirtää alataso mieleiselle paikalle. Alatasoja alkaa tulla näkyviin sitä mukaan enemmän, kun koehenkilö siirtää niitä ylätasojen alle.

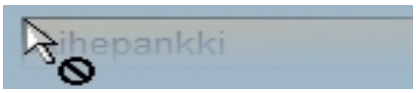


KUVA 8. Esimerkki sovelluksen ulkoasusta

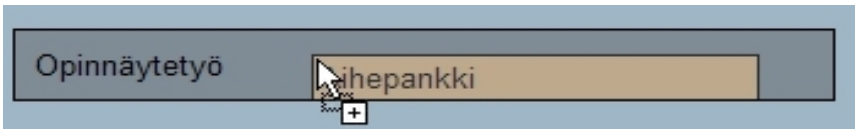
Selain reagoi automaattisesti, jos elementille on annettu draggable-attribuutti ja silloin hiiren kursori muuttuu, kun koehenkilö vie sen kortin päälle. Käyttöjärjestelmä sekä koneen asetukset vaikuttavat siihen, minkä näköinen kursori on. Kortti on mahdollista vetää ja pudottaa, kun hiiren kursori muuttuu kuvan 9 mukaiseksi. Korttia ei voida pudottaa alueelle, jos hiiren kursori muuttuu kuvan 10 näköiseksi. Kortti voidaan pudottaa alueelle, kun hiiren kursori muuttuu kuvan 11 näköiseksi.



KUVA 9. Hiiren kursori, kun drag and drop mahdollista

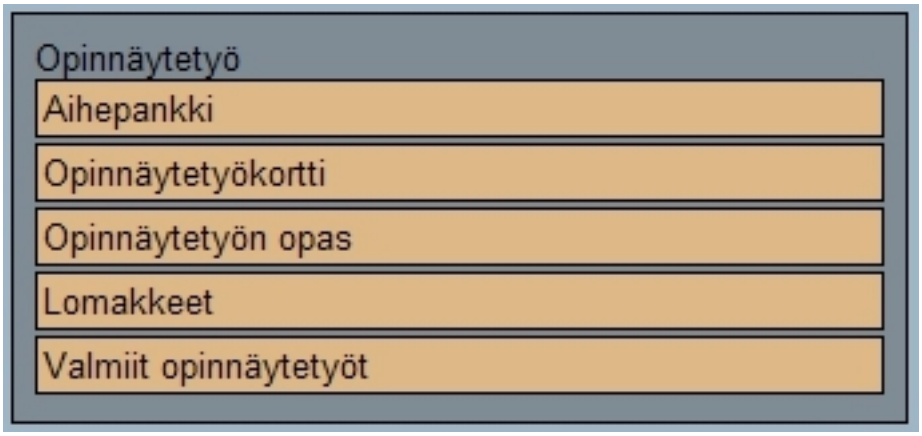


KUVA 10. Hiiren kursori, kun korttia ei voida pudottaa alueelle



KUVA 11. Hiiren kursori, kun kortti on mahdollista pudottaa alueelle

Alatasoja voidaan ylätaso kortin sisällä järjestellä vielä mieleiseen järjestykseen. Kuvassa 12 on esiteltynä esimerkki valmiista rakenteesta. Tiedon tallennuksen yhteydessä kortit tallennetaan sellaisessa järjestyksessä, kun ne on ylätason sisällä. Alatason kortti, joka on ensimmäisenä on koehenkilön mielestä tärkein.



KUVA 12. Esimerkki rakenne

Lopuksi koehenkilö painaa tallenna nappia. Tallennus napin painaminen laukaisee JavaScriptissä funktion, joka käy listan läpi ja rakentaa siitä tietojono. Ajaxin XMLHttpRequest:in GET-metodin avulla tietojono kuljetetaan erilliseen php ohjelmaan, jossa tieto pilkotaan ja viedään tietokantaan. Aikaisemmin jo mainitsin, että saman asian olisi voinut tehdä jQueryllä.

Luvussa 2.6 on kerrottuna mitenkä XMLHttpRequest-objekti toimii. Kuvassa 13 on esiteltynä osa skriptiä, jolla tiedon tallennus tapahtuu. Ylemmän funktion sisällä käydään listaelementit läpi ja tarkastellaan, että missä järjestyksessä ylätasot on ja

mikä alataso on minkäkin ylätason sisällä. Ylä- ja alatasoista luodaan jono, joka liitetään osaksi XMLHttpRequestin urli eli osoite osaa. XMLHttpRequest open-metodiin lisätään GET-metodi, urli (osoite) ja true, joka tarkoittaa, että tieto lähetetään asynkronisesti. Koska tieto lähetetään asynkronisesti, lisätään onreadystatechange tapahtumakäsittelijä, joka laukaisee alemman funktion, kun se on valmis. Alemman funktion saavuttaessa neljännen valmius tilan (readyState=4), ollaan valmiina ilmoittamaan käyttäjälle, että tieto on mennyt eteenpäin ja kiitetään käyttäjää.

```
function save() {
        var kayttaja_id = document.getElementById('kayttaja_id');
        var projektiid = document.getElementById('projektiid');
        var tallenna = "";
        var uls = document.getElementsByTagName('UL');
        for(var i=0;i<uls.length;i++) {
                var lis = uls[i].getElementsByTagName('LI');
                for(var j=0;j<lis.length;j++) {
                if(tallenna.length>0) { tallenna = tallenna + ";"; }
                tallenna = tallenna + uls[i].id + '/' + lis[j].id + '/' + kayttaja_id.value + '/' + projektiid.value
                }
        }
    var urli = "http://localhost/opsaproject/tallenna.php/?tallenna="+tallenna;
    http.open("GET", urli, true);
    http.onreadystatechange = kasitteleHTTP;
    http.send(null);
}
function kasitteleHTTP() {
        if (http.readyState==4 || http.readyState=="complete") {
        document.getElementById('saveContent').innerHTML = 'Tallennus on suoritettu! Kiitos osallistumisesta.';
        }
}
```

KUVA 13. Esimerkki koodi XMLHttpRequest

Tiedot voidaan hakea tietokannasta Excel-taulukkoon (kuva 14). Tämä on muokkaamatonta tietoa ja siksi se ei ole lopullinen. Excel-taulukossa lukee projektinnimi sekä muut projektiin liittyvät tiedot. Ylätasot (Excel-taulukossa päätaso) ja alataso sekä niiden id:t. Käyttäjä numero erottelee koehenkilöt toisistaan. Tieto on valmiina Excel-taulukossa, joten sitä voidaan analysoida tämän jälkeen. Sovellukseen voisi jatkossa kehitellä osion, jossa Excel-taulukon sijaan tiedot näkyisivät suoraan sovelluksessa esimerkiksi pylväskaaviona.

	A	B	C	D	E	F
1	Projektinimi	Student				
2	Id	Päätaso id	Päätaso nimi	Alataso id	Alataso nimi	Käyttäjä nro
3	19	16	Ajankohtaista	412	Aikataulut	3
4	20	16	Ajankohtaista	42	Ajankohtaista paikkakunnittain	3
5	21	16	Ajankohtaista	44	Ilmoitustaulu	3
6	22	16	Ajankohtaista	43	Tapahtumat	3
7	23	16	Ajankohtaista	45	Ruokalista	3
8	24	16	Ajankohtaista	46	Lukujärjestykset	3
9	25	16	Ajankohtaista	41	Mamk uutiset	3
10	26	16	Ajankohtaista	411	Tentit	3
11	27	16	Ajankohtaista	47	Pakolliset ilmoittautumiset	3
12	28	25	Keskustelu	431	Julkaisut	3
13	29	19	Mamk info	417	Kartat	3
14	30	19	Mamk info	418	Osoitteet	3

KUVA 14. Esimerkki Excel-taulukko

5 PÄÄTÄNTÖ

Halusin opinnäytetyöni alkujaan liittyvän jollain tavalla peleihin. Aluksi suunnitteilla oli korttipelin tapainen idea, mutta lopulta se muokkautui sovellukseksi, josta on hyötyä verkkosivujen ja sovellusten tekijöille. HTML5 tarjosi drag and drop-tekniikan, jolla sovellus oli mahdollista tehdä. Mielestäni opinnäytetyöni täytti tavoitteet luoda sovellus, jolla voidaan saada selville paras mahdollinen navigaatiorakenne verkkosivua tai sovellusta varten.

Opinnäytetyötäni tehdessä pääsin hyvin tutustumaan uuteen HTML5:teen ja sen ominaisuuksiin. HTML5:llä uskon tai ainakin toivon olevan hyvä tulevaisuus, kunhan sen standardi saadaan ensin tehtyä loppuun ja kaikki suuremmat selaimet alkavat tukemaan sitä. Itse pidän HTML5:den mukana tulevista ominaisuuksista, kuten helpoista videon ja äänen lisäämisestä osaksi verkkosivua.

Ilman ongelmia työ ei aina sujunut, joten ongelmia kohtasinkin tiedon tallentamisessa, siinä mitenkä saan sen tietokantaan haluamallani tavalla. Aluksi oli ongelmia saada tieto kulkemaan halutulla tavalla eteenpäin ja lopuksi piti vielä keksiä mitenkä pilkon tiedon paloiksi, joka menee sitten tietokantaan. Lopulta kuitenkin sain ratkaistua ongelmani ja sain tiedon tallennuksen toimimaan.

LÄHTEET

Adobe 2011. Yrityksen WWW-sivut. http://www.adobe.com. Ei päivitystietoja. Luettu 11.1.2011.

Asleson, Ryan & Schutta, Nathaniel T. 2007. Ajax – Tehokas hallinta. Jyväskylä: Gummerrus Kirjapaino Oy.

Jackson, Dean & Kesteren, Anne 2006. The XMLHttpRequest Object. WWW-dokumentti. http://www.w3.org/TR/2006/WD-XMLHttpRequest-20060405/. Päivitetty 05.04.2006. Luettu 03.01.2011.

jQuery 2011. Yrityksen WWW-sivut. http://jquery.com. Ei päivitystietoja. Luettu 21.01.2011.

jQuery Project 2011. Yrityksen WWW-sivut. http://jquery.org. Ei päivitystietoja. Luettu 21.01.2011.

Järvinen, Petteri 2010. HTML5 uudistaa webin. Tietokone 4, 48-50.

Kesteren, Anne 2010a. XMLHttpRequest. WWW-dokumentti. http://www.w3.org/TR/XMLHttpRequest/. Päivitetty 03.08.2010. Luettu 03.01.2011.

Kesteren, Anne 2010b. HTML5 differences from HTML4. WWW-dokumentti. http://www.w3.org/TR/html5-diff/. Päivitetty 19.10.2010. Luettu 07.01.2011.

Korpela, Jukka K. & Linjama, Tero 2005. Web-suunnittelu. Porvoo: WS Bookwell.

Microsoft Corporation 2011. Silverlight Overview. WWW-dokumentti. http://msdn.microsoft.com/en-us/library/bb404700%28v=VS.95%29.aspx. Ei päivitystietoja. Luettu 27.01.2011.

Moroney, Laurence 2008. Introducing Microsoft Silverlight 2. United States of America.

Negrino, Tom & Smith, Dori 2007. JavaScript – Teokas hallinta. Jyväskylä: Gummerrus Kirjapaino Oy.

Nielsen, Jacob 2004. Card Sorting: How Many Users to Test. WWW-dokumentti. http://www.useit.com/alertbox/20040719.html. Päivitetty 19.7.2004. Luettu 1.2.2011.

Oliver, Chris 2006. Chris Oliver's Weblog. WWW-dokumentti. http://blogs.sun.com/chrisoliver/entry/f3. Päivitetty 8.11.2006. Luettu 28.01.2011.

Oracle Corporation 2010. Yrityksen WWW-sivut. http://www.javafx.com. Ei päivitystietoja. Luettu 28.01.2011.

Paananen, Petteri 2008. Flash – julkaisijan opas. Saarijärvi: Offset Oy.

Pirinen, Teemu 2009. Card sorting -käytettävyystestausmenetelmä. Tampereen ammattikorkeakoulu. Tietojenkäsittelyn koulutusohjelma. Opinnäytetyö.

Resig, John 2011. Tutorials: How jQuery Works. WWW-dokumentti. http://docs.jquery.com/Tutorials:How_jQuery_Works. Ei päivitystietoja. Luettu 21.01.2011.

Resig, John 2008. jQuery, Microsoft, and Nokia. WWW-dokumentti. http://blog.jquery.com/2008/09/28/jquery-microsoft-nokia/. Päivitetty 28.9.2008. Luettu 21.01.2011.

Robertson, James 2001. Information design using card sorting. PDF-dokumentti. http://www.steptwo.com.au/files/cardsorting.pdf. Ei Päivitystietoja. Luettu 1.2.2011.

Teague, Jason Cranford 2004. DHTML and CSS for the World Wide Web, Third Edition. United States of America.

Uoma Oy 2011. Yrityksen WWW-sivut. http://www.uoma.fi/. Ei päivitystietoja. Luettu 31.01.2011.

W3schools 2010a. JavaScript Tutorial. WWW-dokumentti. http://www.w3schools.com/js/default.asp. Ei päivitystietoja. Luettu 26.11.2010.

W3schools 2010b. DHTML Tutorial. WWW-dokumentti. http://www.w3schools.com/dhtml/default.asp. Ei päivitystietoja. Luettu 15.12.2010.

W3schools 2010c. CSS Tutorial. WWW-dokumentti. http://www.w3schools.com/css/default.asp. Ei päivitystietoja. Luettu 10.12.2010.

W3schools 2011a. HTML Tutorial. WWW-dokumentti. http://www.w3schools.com/html/default.asp. Ei päivitystietoja. Luettu 05.01.2011.

W3schools 2011b. HTML5 Tutorial. WWW-dokumentti. http://www.w3schools.com/html5/default.asp. Ei päivitystietoja. Luettu 05.01.2011.

W3schools 2011c. XML Tutorial. WWW-dokumentti. http://www.w3schools.com/xml/default.asp. Ei päivitystietoja. Luettu 03.01.2011.

W3schools 2011d. CSS3 Tutorial. WWW-dokumentti. http://w3schools.com/css3/default.asp. Ei päivitystietoja. Luettu 19.01.2011.

W3schools 2011e. Ajax Tutorial. WWW-dokumentti. http://w3schools.com/ajax/default.asp. Ei päivitystietoja. Luettu 20.01.2011.

W3schools 2011f. jQuery Tutorial. WWW-dokumentti. http://w3schools.com/jquery/default.asp. Ei päivitystietoja. Luettu 24.01.2011.

Wood, Jed & Wood, Larry 2008. Card Sorting: Current Practices and Beyond. PDF-dokumentti. http://www.upassoc.org/upa_publications/jus/2008november/JUS_Wood_Nov2008.pdf. Päivitetty marraskuu 2008. Luettu 1.2.2011.